IN THE UNITED STATES PATENT & TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS & INTERFERENCES

In re App. No.:	09/449,021	)	PATENT APPLICATION
		)	
Filing Date:	November 24, 1999	)	Art Unit: 2192
		)	
Inventors:	Emmelmann	)	Examiner: C. Kendall
		)	
Title:	*Interactive Server Side Components*	)	
		)	Customer No.: 08685

APPELLANT'S REPLY BRIEF

APPELLANT'S REPLY BRIEF

TABLE OF CONTENTS

STATUS OF CLAIMS

Claims 1-2, 22-23, 26, 30, 32-33, 41-42, 59-63, 67-69 and 71-73 stand finally rejected in the Office Action dated January 21, 2010, and are the subject of this appeal.

In the Examiner's Answer, claims 6, 8, 51-58, 74-96 and 114-128 were identified as allowable, and claims 3-5, 24-25, 27-29, 31, 43, 64-66 and 70 were objected to but identified as allowable. Appellant has therefore requested cancellation of these claims in a separate amendment filed concurrently herewith; appellant has also filed a continuation application in order to capture this allowable subject matter.

Claims 9-21, 34-40, 44-50 and 97-113 have been previously withdrawn as drawn to non-elected subject matter and are formally cancelled in the amendment request. Claim 7 has been previously cancelled.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1-2, 22-23, 26, 30, 32-33, 41-42, 59-63, 67-69 and 71-73 are unpatentable under Section 103(a) over the combination of the 1996 article by Crespo and Bier entitled: *WebWriter: A Browser-Based Editor for Constructing Web Applications* ("WebWriter I") and the 1997 article by Crespo, Chang and Bier entitled *Responsive Interaction for a Large Web Application: The Meteor Shower Architecture in the WebWriter II Editor* ("WebWriter II").

ARGUMENTS

I. INTRODUCTION

Appellant appreciates the indication of allowable subject matter on pp. 3-4 of the Examiner's Answer. Appellant has therefore filed an amendment requesting cancellation of many of the pending claims on appeal thereby greatly simplifying the appeal. Thus, instead of 10 independent claims, only 4 independent claims would remain on appeal. A complete listing of the claims on appeal if the amendment is entered is included as Appendix A. Concurrently, appellant has filed a continuation application to pursue those allowable claims in a separate application.

The now pending claims on appeal relate to appellant's development of an improved editor for editing of server side dynamic web applications. The editor itself is a web application running on a server and in a web browser. Appellant's editor runs edited applications during editing, i.e., the editor operates on documents that have been generated by a document generator running the application being edited. (see the detailed discussion in section A.i, pp. 13-15 of appeal brief). This beneficially shows the content of dynamically generated parts of the documents during editing while the cited prior art just shows placeholders, not the content (see the detailed discussion in section A.ii, pp. 15-18 of Appeal Brief, esp. p. 16, 3rd and 4th paragraphs, and p. 17). This helps the developer understand exactly what the user sees, and he can effectively and efficiently make adjustments to document templates to reflect that better understanding of what the user sees.

In appellant's view, the Examiner continues to misunderstand an article describing a system named WebWriter consisting of two separate programs, the WebWriter Editor, and the WebWriter Page Generator. The Examiner seems to believe that the Editor and the Page Generator of the WebWriter system somehow cooperate in the editing process (see the Answer, Response b, p. 22), but they do not, and there is nothing in either WebWriter article to describe or support such a view. In fact, it supports the opposite view, as appellant has shown (see the detailed discussion in

section C.i, pp. 40-32 of the Appeal Brief, and section IV.C below). Indeed, the Examiner's view that the WebWriter Editor and the WebWriter Page Generator as described cooperate in some way in the WebWriter system has led to unclear statements of rejection. For example, the Examiner puts the WebWriter Page Generator into correspondence with the claimed editor (p. 22 in the Answer, sec. III below), but then argues using other portions of the article that describe the WebWriter Editor and not the WebWriter Page Generator (see section IV.C below, e.g. p. 26 in the Anwser).

However, the WebWriter Editor generates pages during editing, without actually executing the edited application, and without using the WebWriter Page Generator (see section IV.D, 3rd para. below, and the first full para. on p. 41 in section C.i of the Appeal Brief). As described in the WebWriter article, after the edited application has been saved, it can be executed using the WebWriter Page Generator (sec A.ii p. 16, 1st para of Appeal Brief).

The Examiner also continues to mistakenly believe that the WebWriter Editor executes the application being edited during editing. This is discussed as argument (b) in the Examiner's Answer and in this Reply Brief in section IV.B below. The Examiner also discusses appellant's argument (d), that the editor program operating on the generated documents via editing features was not addressed in the final rejection. However, the Examiner appears to ignore that the term "**the** generated documents" is defined clearly in the claim (as documents being generated by the document generator, which is running at least part of the application being edited) and so misunderstood appellants argument and just argues that the editor operates one some generated documents (see sec. IV.D below).

The Examiner also cited a second article, WebWriter II, mainly for the dependent claims; for the independent claims, the Examiner uses it only as prior art for a "server." (see p. 6 of the Answer). However, WebWriter II is not an extension of WebWriter I but uses a totally different architecture called "meteor shower" and therefore a combination is improper, as detailed in section C.iii, p. 43 of the Appeal Brief.

This Reply Brief mainly discusses the responses to arguments in the Examiner's answer. It does not give a complete introduction and also does not repeat all the arguments given in the appeal brief, but provides citations to those previous discussions. Appellant notes that there are also main arguments discussed in the appeal

brief, as noted below, that have not been discussed by the Examiner in his Answer, and appellant therefore asks the Board to carefully consider these arguments in the Appeal Brief as well.

II. THE PRIOR ART DOES NOT TEACH A CONNECTION FROM THE WEBWRITER PAGE GENERATOR TO THE EDITOR

A. The Examiner Has Not Proven A Connection From The WebWriter Page Generator To The WebWriter Editor

The Examiner has failed to address any of the well-reasoned arguments given in the sections A.i and A.ii on pp. 13-18 of the Appeal Brief about why the Examiner is mistaken regarding the true nature and scope of the cited prior art, and likewise in sections C.i, C.ii and C.iii on pp. 40-43 of the Appeal Brief, which discuss the teachings of the cited prior art in great detail.

Since interpretation of the cited prior art is a key issue in this appeal, and likewise this affects how the claims are construed, it is important that the prior art references be properly considered and referenced for all they do and do not teach. The Examiner's failure to rebut much of appellant's reasoning regarding the prior art suggests there is no adequate rebuttal.

The WebWriter System has two distinct and separate programs: the WebWriter Editor and the WebWriter Page Generator. The WebWriter Editor creates and/or edits web applications, then saves the applications. The WebWriter Page Generator is used to run the saved applications. The WebWriter Editor, however, does not run the edited application and it also does not use the WebWriter Page Generator to run the edited application. The WebWriter Editor generates pages on its own without running the application being edited and without the help of the WebWriter Page Generator.

B. Appellant Has Provided Substantial Evidence To Show That There Is No Connection From The WebWriter Page Generator To The WebWriter Editor

Claim 1 requires a document generator program running at least part of one of the applications being edited. The documents generated by the document generator

program are required to include additional editing features. In addition, claim 1 requires an editor working on the generated documents via the editing features (*see* appeal brief sections B.i.b and B.i.d, p. 19).

The WebWriter article teaches away from those features by describing that the WebWriter Editor generates pages (i) **without** the help of the WebWriter Page Generator and (ii) without executing the edited application, and not describing that the WebWriter Page Generator generates pages having editing features. The Appeal Brief includes many citations and arguments that are persuasive and result in the conclusion that WebWriter works this way:

- The article talks about "inserting of computed content" and explicitly adds "at run-time" which at least suggests that computed content is not inserted at editing time (*see* Appeal Brief at sec. A.i, p.14);
- The article says "Once an application is … saved to disk it can be run" (*Id.* at sec. A.ii, p. 16, 1st para.);
- The WebWriter editor has its own page generation and so there is no need for using the WebWriter Page Generator (*Id.* at sec. A.ii, p. 16, 2nd para; sec. C.i, pp. 40-42; also see section IV.D below);
- During editing, the edited page template is represented as a tree while the page generator needs a text file (*Id.* at sec. A.ii, p. 16, 2nd para; sec. C.ii, p. 42);
- The article describes and shows placeholders for dynamic areas, which indicates that they are not filled during editing but only at runtime (A.ii, p. 16, 3rd and 4th paras., p. 17).

III. THE CLAIM REJECTIONS ARE INADEQUATE

The Examiner copied the text of the final rejection into the Examiner's Answer, and the final rejection is discussed in detail in the Appeal Brief in section B.

Appellant finds the Examiner's rejection to be inadequate. The key to supporting any rejection under 35 U.S.C. 103 is the clear articulation of the reason(s) why the claimed invention would have been obvious. *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398, 82 USPQ.2d 1385, 1396 (2007); MPEP 2141. The Examiner has failed to provide well articulated reasoning describing a clear correspondence between the elements in the

claim and the cited prior art. In the Appeal Brief, appellant mentioned various elements in the claims that had not been addressed by the Examiner. Rather than improve upon the rejection, the Examiner has addressed *some* of these points in the form of arguments in the Examiner's Answer which is not sufficient to demonstrate obviousness. The answers provided to appellant's arguments are not convincing and sometimes do not even answer the right question, as discussed in depth below. The Examiner has failed to demonstrate obviousness, particularly in light of appellant's showing to rebut the Examiner's misunderstanding with regard to the teachings of the cited prior art.

The rejection of claim 1 (which the Examiner uses in his reasoning about all the independent claims) does not address the main distinctive features discussed in section II above, e.g., the limitation of "operating on **the** generated documents." Appellant wonders, after reading response (d) in the Answer, whether the Examiner just ignores the definite article "the" in front of "generated documents". However, the use of the definite article clearly refers back to the antecedent use of generated documents in the claim, i.e., the generated documents generated by the document generator program. (*See* section B.i.d of the appeal brief and sections IV.D, IV.B and IV.E of this Reply Brief.) Editing features, including the fact that the generated documents must include them, and that the editor operates via theses editing features, are also not discussed in the final rejection, but appellant discusses this important limitation in section B.i.b and B.i.d of the Appeal Brief and section IV.D of this Reply Brief. Appellant does not deny that there are editing features used by the WebWriter Editor, but these features are not inserted into the generated documents by the WebWriter Page Generator (as required by the claim), but only into a page generated by the WebWriter Editor. This is discussed in more detail in the argument section IV.D below.

The Examiner has repeatedly cited "page 9, column 1, see Web Writer Page Generator" of the WebWriter article when describing the association of the claimed editor with WebWriter's Page Generator (see Examiner's Answer at 22 and 6), although he has never explicitly discussed that he is making this association, and this seems odd – one might expect instead that the WebWriter Editor be associated with the claimed editor. However, the Examiner also has not refuted appellant's arguments against such an association. In fact, it remains unclear to appellant exactly how the Examiner reads the prior art to correspond with the claims. As the rejection currently reads the

WebWriter Editor is not recited at all, but on the other hand the Examiner argues about the WebWriter Editor and recites portions of the article that discuss the WebWriter Editor and not the WebWriter Page Generator.

(i) Citing WebWriter's Page Generator as prior art for the claimed editor fails for the simple reason that the WebWriter Page Generator is not an editor at all. This is discussed in argument section IV.C below. It also fails because the Examiner recites just one program, the WebWriter Page Generator, as the relevant prior art corresponding to two different portions of the claim, the claimed document generator and the claimed editor. Also, the WebWriter Page Generator does not operate via editing features as required for the claimed editor in claim 1, and discussed in section IV.D below.

(ii) Assuming appellant misunderstood the Examiner's reasoning, and that it was the Examiner's intention to cite the WebWriter Editor as prior art for the claimed editor, then the logic still fails, since WebWriter Editor and WebWriter Page Generator are not connected in the way claimed by appellant, as discussed in argument sections IV.B and IV.D below. The WebWriter Editor is not operating on the pages generated by the WebWriter Page Generator, but it is generating pages on its own.

IV. THE EXAMINER'S REBUTTAL TO ARGUMENTS IS NOT CONVINCING

A. Introduction

The Examiner discusses only three of appellant's arguments in the Answer, described as arguments (b), (c), and (d), which are apparently intended to correspond to subsections b), c) and d) of section B.i of the Appeal Brief, which discusses only claim 1 starting on p. 19. These sections in the Appeal Brief, however, contain various other arguments as well, and it appears that the Examiner just selected one argument per section to discuss. Appellant thinks that the other arguments are also valid and provide additional support for appellant's position.

B. The Examiner Is Not Persuasive In Arguing That The Web Writer Editor Permits Editing Of A Running Application

Regarding Argument (b), the Examiner maintains that WebWriter's editor runs the edited application during editing. Appellant strongly disagrees.

On page 22 of the Answer, the Examiner merely repeats the rejection of claim 1 from the final rejection. Appellant fails to see the relevance. The citations provided by the Examiner for both the document generator and editor claim elements explicitly recite the WebWriter Page Generator, not the WebWriter Editor, while argument (b) deals with the WebWriter Editor. For all discussion of the claim 1 rejection, appellant refers to section B.i on pp. 18-24 of the Appeal Brief, and more specifically, subsection c discussing the Examiner's reasoning.

On pp. 23-24, the Examiner reproduced three sections from the WebWriter I article, and then argued that the WebWriter Page Generator runs at least part of the application. Argument (b), however, discusses the WebWriter Editor and not the WebWriter Page Generator. The WebWriter Editor and WebWriter Page Generator are described as separate programs that are useful as part of an integrated system for creating server-based web applications. But unlike appellant's claims, there seems to be no data flow in WebWriter from the WebWriter Page Generator to the Editor that permits editing a running application in the Editor.

The Examiner seems to simply assume without argument that because the WebWriter Page Generator has a certain feature (for example, replacing dynamic content at runtime), the WebWriter Editor has the same feature. However, the WebWriter article does not support such a conclusion. The WebWriter Page Generator is neither part of nor is it being used by the WebWriter Editor, and nothing in the cited articles would support such a conclusion, as appellant has already described in great detail in the Appeal Brief in section A.i (p. 5), section A.ii (p. 7), and section C.i (p. 40). Thus, there appears to be no support for the Examiner's statement (b) that the WebWriter Editor runs the application during editing.

The Examiner further argues that WebWriter shows "the application is running while at least editing a part of the application during runtime." (See Answer at 24). Appellant respectfully disagrees. The Examiner cites a portion of the WebWriter article,

namely: "Dynamic areas are inserted into a template page to indicate regions of the page that will be filled in at runtime." However, a simple parsing of this sentence reveals two separate events: (1) dynamic areas are inserted into a template page to indicate regions of the page; and (2) the regions will be filled in at runtime. In appelant's reading, neither (1) nor (2) support the Examiner's statement:

The phrase "at runtime" only refers to the filling of the regions, and not to the inserting of dynamic areas, which indicates (when viewed in the context of the cited article) that events (1) and (2) take place at different times and event (2) takes place only at run-time. The "at runtime" clause makes clear that the filling indeed takes place at runtime and puts it in contrast with the inserting event. So, because event (1) does not take place at run-time, phrase (1) does not support the Examiner's statement "the application is running while at least editing a part of the application during runtime."

The filling event (2) of the regions at runtime is described in WebWriter I on p. 9, left column, heading "The Web Writer Page Generator." As discussed in sections C below and B.i.c of the Appeal Brief, the WebWriter Page Generator is not editing template pages and consequently the filling is also not an editing operation. Thus, the Examiner's statement "that WebWriter shows the application is running while at least editing a part of the application during runtime" is also not supported by the second part (2) of the citation.

Even more importantly this does not refute appellant's real argument (b) as cited in the Examiner's Answer on p. 21 that the WebWriter Editor does not run the application being edited because (2) applies to actions of the WebWriter Page Generator. As discussed above, the WebWriter Page Generator working at runtime is separate from the WebWriter Editor, and reasoning about the WebWriter Page Generator does not necessarily say or imply anything about the WebWriter Editor.

The Examiner further reasons on page 25 that the editor program claimed is also disclosed under the heading "placing dynamic areas" by the statement "which clearly discloses areas of the code which are dynamically replaced at runtime." In appellant's reading, this statement does not disclose editing of template pages, but template expansion that temporarily fills out a template and sends the result to the client computer. This is discussed in the Appeal Brief on page 20, section B.i.c. Of course, the

WebWriter article discusses the WebWriter editor in depth, but the cited portion seems to discuss template expansion of the WebWriter Page Generator.

Arguments of Appeal Brief section B.i.b not discussed

With respect to the question of running during editing, the Examiner gave some new citations; however, he did not discuss any of appellant's arguments or citations given in section A.i, A.ii and C.i of the Appeal Brief that show that WebWriter does not provide this feature.

Section B.i.b argues that claim 1 should be considered patentable because it requires an editor that operates on the generated documents, whereby the generated documents refers back to documents generated by the document generator.

Also section B.i.b argues that claim 1 should be considered patentable because it requires a document generator generating documents that include additional edition features. Editing features are not mentioned in the section of the article describing the WebWriter Page Generator and also it makes no sense to include editing features into pages generated by the WebWriter Page Generator, because it runs after editing.

C. The Examiner Is Not Persuasive In Arguing That The Web Writer Page Generator Edits Applications

Regarding Argument (c), the Examiner apparently misunderstood appellant's argument and did not reproduce it correctly. The Examiner wrote "Appellant argues on page 20 that the prior art does not disclose any editing functions ..." (See Answer at 25). However, appellant in fact argued that the section entitled "Web Writer Page Generator, which describes the WebWriter Page Generator but was cited by the Examiner against the editor portion of the claim, does not describe any editing functions. (See Appeal Brief at 20). The Examiner appears to believe that the WebWriter Editor and the WebWriter Page Generator are interchangeable, but there is no support in the article itself for any such conclusion.

The Examiner then shows portions from page 9 of the article but from the previous section entitled "The WebWriter Editor" (starting on page 8, column 2). This does not contradict appellant's argument that was referring to the next section labeled

"The WebWriter Page Generator" and therefore appellant still is convinced that his argument as phrased in the Appeal Brief is correct.

To put this in a broader context: The first section of the WebWriter article entitled "The WebWriter Editor" describes the editor working at editing time, and the section entitled "The WebWriter Page Generator" describes the Page Generator working at run-time. The first section "The WebWriter Editor" describes some page generation as well but not running of the edited application which appellant takes as an indication that the WebWriter Editor does not run the edited application during editing. The second section "The WebWriter Page Generator," in contrast, describes another kind of page generation that runs an application, but does not describe editing features.
In addition, there are further portions of the article showing that the WebWriter Page Generator is not an editor. The first paragraph of the section "The WebWriter Page Generator" says clearly

"*When a WebWriter application is running each new page is assembled by the WebWriter Page Generator, another server based CGI C++ program that knows how to read a template page, run the scripts specified in that template page and create a new page for display to the user*."

So the WebWriter Page Generator is not modifying a template page into a different template page as an editor would do, but it transforms a template page into a page (but not into a template page) for display to the user. It therefore does not edit applications a required by the claim. Also, in the abstract of the WebWriter I article, the WebWriter Editor and the WebWriter Page Generator are contrasted "*WebWriter includes a direct manipulation Web page editor*"..."*and the WebWriter Page Generator, which creates new pages as the application runs*" which makes clear that the WebWriter Page Generator creates new pages while the WebWriter Editor edits template pages.

Appellant has provided various additional arguments for his position in section B.i.c (p. 20) of the Appeal Brief.

Arguments of section B.i.c not discussed

Section c discusses the Examiner's specific mentioning of the term "on the fly" as used in the WebWriter article.

D. The Examiner Is Not Persuasive In Arguing That Webwriter's Editor Operates Via **THE** Editing Features

Regarding Argument (d), beginning on p. 26, the Examiner discusses the editor program operating via editing features; however, appellant's claim language and argument refers to "operating via **the** editing features".

Subsection B.i.d, p. 21 of the Appeal Brief, specifically refers to the claim limitations of the "editor program operating on **the** generated documents" and "via **the** editing features" (emphasis added) and makes clear that these two limitations connect the running application to the editing process. The use of the definite article "**the**" in the recited claim language makes clear that the editor operates on documents being generated by the document generator, and also that the editor operates via **the** editing features that appear in the documents generated by the document generator.

Even the Examiner cites material from the WebWriter article that supports appellant's position. On pp. 26-27, the Examiner cites material describing the editor: "The WebWriter Editor is a server based CGI service . . . **it** generates a two column page . . . Each page **generated by the Editor** ..." (emphasis added). This portion of the article makes it crystal clear that in the WebWriter system, it is the WebWriter Editor itself that generates pages for use during editing. In contrast, the claim requires the editor to operate on the documents generated by the document generator.

Appellant notes that in response (d), the Examiner did not mention editing features, in particular, he did not identify what items described in the WebWriter article correspond to editing features. Further, he also did not argue that the editing features are contained in **the** generated documents, as required by the claim.

E. The Examiner Is Not Persuasive In Arguing That Webwriter Editor Runs The Edited Application

On page 27 of the Answer, the Examiner writes "regarding Appellants argument in the same section on page 23 first paragraph." However, appellant notes that section d) ends on page 21 and so the Examiner is probably referring to argument section e).

On page 28, the Examiner reasons "it is disclosed above that the dynamic areas are inserted with content at runtime by a program." However, this is not what was actually written in the article. The article says "Dynamic areas are inserted into a template page to indicate regions of the page that will be filled in at runtime by a program." This statement was already discussed in detail in section B, p. 9, 5th para. and following. The filling in of dynamic areas is an operation performed by the WebWriter Page Generator as described in the WebWriter article on page 9 under the heading "The WebWriter Page Generator." As previously discussed, this does not say anything about the WebWriter Editor and so does not refute appellant's argument that the WebWriter Editor does not run the application during editing.

Note that appellant further argues that because the WebWriter Editor does not run the application during editing, the WebWriter Editor cannot possibly be operating on the pages generated by the WebWriter Page Generator as claimed. This reasoning relies on the argument that the WebWriter Editor does not run the application, and whether the WebWriter Page Generator performs some kind of editing is irrelevant for that reason.

F. The Rest of the Independent Claims

1. Claim 22

The Examiner states that claim 1 and claim 22 differ only with regard to the inclusion of components. Appellant disagrees, and notes that there are more differences, e.g., editing features or execution of applications in claim 1 versus execution of components in claim 22. Appellant also refers to his reasoning for separate patentability in the Appeal Brief and notes that this claim should be treated separately.

Appellant thinks that claim 22 is patentable over the cited art because, as in claim 1, it requires the editor to operate on documents that were generated by the document generator program. This is expressed using the claim language "an editor program operating"... "**on generated documents**"... "and a document generator program"... "for generating **the generated documents**". In contrast in WebWriter it seems to be the WebWriter editor that generates the pages the editor operates on, not the WebWriter page generator, as discussed in sections IV.C and IV.D above and section A.ii and C.i of the Appeal Brief.

The Examiner still did not include any specific reasoning for claim 22 but instead used the reasoning for claim 1 of the final rejection reproduced on page 5-6 of the Answer where he only cites the WebWriter Page Generator and not the WebWriter Editor (as discussed above in section III and section IV.C).

The Examiner's argument's on page 29 discuss the WebWriter Page Generator, while the citation given on page 29 that purports to support these arguments comes from the implementation section "The WebWriter Editor" on page 9 and therefore describes the WebWriter Editor and not the WebWriter Page Generator. So, the Examiner's arguments are not supported by the prior art with regard to the WebWriter Page Generator. As discussed in section IV.C, the WebWriter page generator just does template expansion and is unrelated to editing.

It would have seemed more natural to appellant if the Examiner had cited the WebWriter Editor and the WebWriter Page Generator together. However, as explained above, claim 22 overcomes that combination by requiring the editor to operate on the documents generated by the document generator, while the WebWriter Editor generates pages on its own without the WebWriter Page Generator as discussed in section IV.D above. The theoretical third alternative of interpreting the WebWriter Editor as document generator fails, because the claim requires the document generator to execute components, and the WebWriter Editor does not execute the dynamic areas, only the WebWriter Page Generator does that.

On page 29 of the Answer, the Examiner also lists various items he interprets to be components, without addressing the limitation in claim 22 requiring that the document generator has instructions for executing the components. So, for example, buttons and check boxes are not components in the context of the claim because they are being executed by the browser and not by the document generator.

Thus, appellant's arguments remain valid and claim 22 is for these and the other reasons discussed in the Appeal Brief patentable over the cited art.

2. Claim 26

Appellant refers to his reasoning for separate patentability in the Appeal Brief and notes that therefore this claim should be treated separately.

Claim 26 requires that the second document appear and function similar to the run-time view of the first document. In his reasoning on page 30, the Examiner recites a portion of the WebWriter article: "Each page generated by the Editor contains HTML code that represents the appearance of the current document in the left column." There is a big difference between the word "represent" used in the WebWriter article and the word "similar" used in the claim. In fact, as discussed in section A.ii on pp. 16-17 of the Appeal Brief, dynamic areas in the HTML document are replaced by placeholders during editing. Although these placeholders represent or stand for the dynamic areas, they do not have a similar appearance, as shown in the figures in the WebWriter article, reproduced in section A.ii of the Appeal Brief. Thus, the cited portion does not support the Examiner's argument.

The Examiner does not discuss the limitation "function similar" which is also part of the argument and required by the claim as well. In fact, the cited portion explicitly talks about "appearance" and not about function. It also only talks about HTML and not about any scripts or similar items with a function. Thus, it appears that the Examiner did not consider the "function similar" part of the argument.

Appellant's argument remains valid and claim 26 is for this and the other reasons discussed in the Appeal Brief patentable over the cited art.

3. Claim 59

The Examiner did not include specific reasoning or specific arguments with respect to claim 59. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claim 59. This claim should be considered separately.

G. The Dependent Claims

With respect to claims 2, 61 and 67-69, the Examiner refers to his reasoning for other claims, without actually discussing appellant's arguments about of grouping of these claims in the Appeal Brief. With respect to claim 41, 42 and 63, the Examiner did not include any arguments in the Examiner's Answer.

Appellant submits that all these claims should be considered separately (except for the group of claim 41 and 42) for the reasons given in the appeal brief and maintains that these claims are patentable over the cited art for the reasons given in the appeal brief.

1. Claim 2

The Examiner did not include specific arguments with respect to claim 2 but referred to claims 1, 22 and 26. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claim 2 and submits that this claim should be considered separately. Appellant still maintains his reasoning provided for claim 2 in the Appeal Brief.

2. Claim 23

The Examiner states that he already addressed editing mode as part of editing. The claim, however, requires running an application in editing mode as discussed in the Appeal Brief, and this issue has not been squarely addressed by the Examiner.

3. Claim 30

The Examiner refers to scripts in WebWriter's template pages. However, reading base claim 26 together with claim 30 requires scripts in the second document, i.e., the document that is generated (see Appeal Brief page 46 for details):

"... transforming at least one first document retrieved from the document store into a second document having **features** ..." (claim 26) and "**the features** include scripts" (claim 30).

The Examiner did not include reasoning about which documents contain the scripts. In addition, claim 26 requires "features which permit editing of the first document." Since claim 30 requires "**the features** include scripts," the Examiner has

failed show that the scripts described in the WebWriter article permit editing of the first document.

4. Claims 41-42

The Examiner did not include specific arguments with respect to the group of claims 41-42. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claims group 41-42. Appellant still maintains his reasoning provided for claims 41-42 in the appeal brief.

5. Claim 60

On page 32 of the Answer, the Examiner states that Webwriter II is an extension of Webwriter I, without providing any support for that statement and without discussing section C.iii on page 34 of the Appeal Brief that addresses this issue.

In appellant's reading, Webwriter II is not at all an extension of Webwriter I, but it is a new implementation using a different architecture (*See* WebWriter II, sec. 1.2 entitled "Increasing interactive performance": "*This paper describes the architecture of a new version of the editor (the WebWriter II Editor) that overcomes these limitations. In this architecture, which we call the meteor shower application architecture both the web browser and the web server collaborate in executing the WebWriter II Editor*").

Because WebWriter II is not an extension of WebWriter I but uses a different architecture, the Examiner's argument is not relevant and therefore void.

6. Claim 61

The Examiner did not include specific arguments with respect to claim 61 but referred to claims 1, 22 and 26. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claim 61 and submits that this claim should be considered separately. Appellant still maintains his reasoning provided for claim 61 in the Appeal Brief.

7. Claim 63

The Examiner did not include specific arguments with respect to the claim 63. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claim 63. Appellant still maintains his reasoning provided for claim 63 in the Appeal Brief.

8. Claim 67

The Examiner did not include specific arguments with respect to claim 67 but referred to claim 1, 22 and 26. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claim 67 and submits that this claim should be considered separately. Appellant still maintains his reasoning provided for claim 67 in the Appeal Brief.

9. Claim 68

The Examiner did not include specific arguments with respect to claim 68 but referred to claim 1, 22 and 26. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claim 68 and submits that this claim should be considered separately. Appellant still maintains his reasoning provided for claim 68 in the Appeal Brief.

10. Claim 69

The Examiner did not include specific arguments with respect to claim 69 but referred to claim 1, 22 and 26. Appellant therefore refers to the Appeal Brief including the reasoning for the separate patentability of claim 69 and submits that this claim should be considered separately. Appellant still maintains his reasoning provided for claim 69 in the Appeal Brief.

11. Claim 72

On page 33 of the Answer, the Examiner interpreted the claim phrase "reload in the browser" as equivalent to save and loads of templates in the WebWriter editor. However, "reload" is a specific function of a web browser. It does not reload the browser, as the Examiner wrote, but a web page displayed by the browser. Appellant believes that

the phrase "reload in the browser" as used in claim 72 cannot be interpreted any different than the standard reload function supplied by a web browser, especially since reload is used in the description that same way. Therefore, the Examiner's argument is void.

12. Claim 73

In the Appeal Brief, appellant argued that claim 73 should be considered separately because it requires the important distinctive feature of displaying without requesting that the document generator generates a document. The Examiner did not address this feature, but just referred to his previous arguments in claims 1, 22 and 26 for "displaying" and "viewing edited content," which is not convincing. Appellant therefore believes that claim 73 should still be considered separately and that it is patentable over WebWriter of this feature and the other reasons given in the Appeal Brief. The claim is patentable over WebWriter II because the base claim was rejected on WebWriter and the combination is improper as discussed with regard to Claim 60 and in section C.iii of the Appeal Brief.

Respectfully submitted,

Date: December 13, 2010 By: /Richard A. Nebb/
Richard A. Nebb
Reg. No. 33,540

DERGOSITS & NOAH LLP
3 Embarcadero Center, Suite 410
San Francisco, California 94111
Telephone: 415.705.6377
Facsimile: 415. 705.6383
Email: rnebb@dergnoah.com

CLAIMS APPENDIX

Claims on Appeal

1. A computer-readable medium encoded with computer programs having executable instructions for editing software applications that run on a data network which couples a server computer and a client computer, wherein the client computer runs a browser program, and whereupon request by the browser program, at least one of the applications generates generated documents for display by the browser program on a display device and responds to the request with the generated documents, comprising:

a document generator program running at least part of one of the applications being edited and generating the generated documents, said generated documents including additional editing features for interpretation by the browser program; and

an editor program dynamically operating on the generated documents displayed by the browser program via the editing features.

2. A computer-readable medium as in claim 1, further encoded with a plurality of components, and wherein the software applications comprise at least one document template capable of containing components, and wherein the editor provides features to insert, modify and delete a component on at least one document template, and wherein the document generator executes selected components on document templates.

22. A computer-readable medium encoded with computer programs having executable instructions to edit and maintain applications using a web browser, comprising:

an editor program operating within the web browser on generated documents and having instructions for inserting, deleting, and modifying components on document templates; and

a document generator program having instructions for processing document templates, for executing said components, and for generating the generated documents from the document templates that are understandable by the web browser.

23. A computer readable medium as in claim 22, wherein the editor program operates a functional application in an edit mode permitting editing of said application directly in the web browser.

26. A system having a data network which couples a server computer to a client computer, the server computer running an application to modify dynamic documents on the server computer, the server computer comprising:

a document store;

a first software program including instructions for transforming at least one first document retrieved from the document store into a second document having features which permit editing of the first document such that at least a part of the second document appears and functions similar to the run-time view of the first document; and

a second software program including instructions to receive information from the client computer and instructions to modify the first document stored in the document store.

30. The system of claim 26, wherein the features include scripts.

32. The system as in claim 26, wherein the features incorporate information regarding the first document into the second document.

33. A system as in claim 32, wherein the information incorporated into the second document is used on the client computer in order to send change requests for the first document to the server computer.

41. A computer-readable medium as in claim 1, the editor program comprising a client part for execution on the client computer.

42. A computer-readable medium as in claim 41, wherein the client part comprises instructions for execution during editing that are automatically downloaded from the server computer in a request prior to editing.

59. A software development system having at least one computer running an application for developing dynamic web documents, said dynamic web documents operating by being transformed into an end user's view upon a request by a web browser, the end user's view being provided to the browser for display on a display device in response to the request, comprising:

an editor program having instructions for dynamically editing dynamic web documents,

a document generator program having instructions for generating generated documents from dynamic web documents which look and function similar to the end user's view of the documents with the addition of editing features,

the editor program comprising first instructions for requesting that the document generator program processes a dynamic web document during editing thereby resulting in a generated document,

the system comprising second instructions for displaying at least some information items contained on said generated document in a view which allows the user to select an item to which a modification function will be applied,

the editor program comprising third instructions to modify the dynamic web document to perform said modification function.

60. The software development system of Claim 59 comprising a data network which couples a server computer and a client computer, the document generator program running on the server computer, the editor program at least partly running on the client computer.

61. The software development system of claim 60 comprising fourth instructions for execution during the document generation to collect edit-information for use by the editor program.

62. The software development system of claim 60, wherein the editor program uses a web browser for displaying said view.

63. The software development system of claim 60, comprising instructions for automatically repeating the request that the document generator processes the dynamic web document when required.

67. The software development system of claim 59, wherein said view looks, except for editing features, similar to the end-user view of the generated document.

68. The software development system of claim 59 comprising sixth instructions to collect edit-information for use by the editor program, said sixth instructions for execution during the document generation.

69. The software development system of claim 68, wherein the editor program uses the edit-information to correctly modify the dynamic web document.

71. The software development system of claim 59, wherein the editor program uses a web browser for displaying said view.

72. The software development system of claim 71, wherein the first instructions comprise seventh instructions for initiating a reload in the browser.

73. The software development system of claim 59 wherein the editor program comprises eighth instructions to display information on at least one element of at least one dynamic web document, that is replaced during document generation, without requesting that the document generator program generates a document.